D3: A Collaborative Infrastructure for Aerospace Design

Joan Walton, Robert E. Filman*, Chris Knight, David J. Korsmeyer, and Diana D. Lee†
NASA Ames Research Center, MS 269-2
Moffett Field, CA 94035
* Research Institute for Advanced Computer Science
† Science Applications International Corporation
{jdwalton | rfilman | cknight | dkorsmeyer | ddlee}@mail.arc.nasa.gov

Introduction

DARWIN is a NASA developed, Internet-based system for enabling aerospace researchers to securely and remotely access and collaborate on the analysis of aerospace vehicle design data, primarily the results of wind-tunnel testing and numeric (e.g., computational fluid-dynamics) model executions. DARWIN captures, stores and indexes data; manages derived knowledge (such as visualizations across multiple datasets); and provides an environment for designers to collaborate in the analysis of test results. DARWIN is an interesting application because it supports high-volumes of data, integrates multiple modalities of data display (e.g., images and data visualizations), and provides non-trivial access control mechanisms. DARWIN enables collaboration by allowing not only sharing visualizations of data, but also commentary about and views of data.

We are currently developing D3, the third generation of DARWIN [8]. Earlier versions of DARWIN were characterized by browser-based interfaces and a hodge-podge of server technologies: CGI scripts, applets, PERL, and so forth. But browsers proved difficult to control, and a proliferation of computational mechanisms proved inefficient and difficult to maintain. D3 substitutes a pure-Java approach for that medley: A Java client communicates (though RMI over HTTPS) with a Java-based application server. Code on the server accesses information from JDBC databases, distributed LDAP security services, and a collaborative information system (CORE, a successor of PostDoc [1].) D3 is a three tier-architecture, but unlike "E-commerce" applications, the data usage pattern suggests different strategies than traditional Enterprise Java Beans—we need to move volumes of related data together, considerable processing happens on the client, and the "business logic" on the server-side is primarily data integration and collaboration. With D3, we are extending DARWIN to handle other data domains and to be a distributed system, where a single login allows a user transparent access to test results from multiple servers and authority domains.

Background

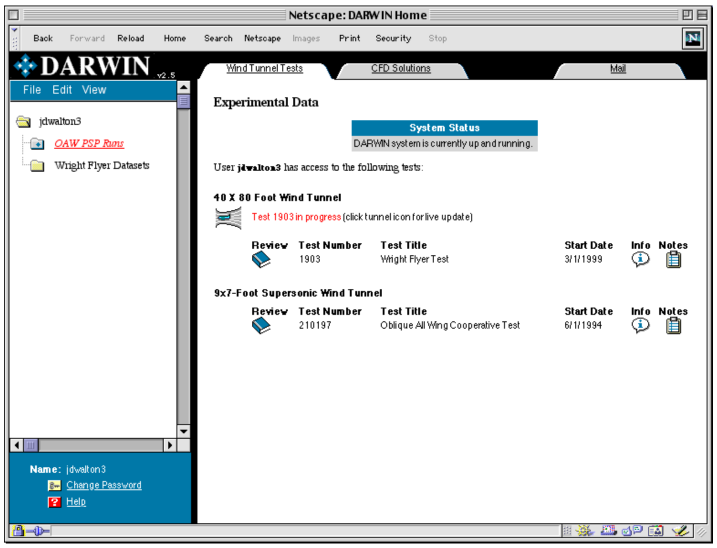
Aerospace designers and engineers use DARWIN to understand and remotely access the results of experimental testing and numerical model analyses of aerospace vehicle designs, principally aircraft in NASA's wind tunnels. Wind tunnel tests place a physical model of a proposed aircraft in an enclosed space, blow 100-600 mile-per-hour winds over the surface of that model, and measure performance attributes such as lift, drag and yaw. Sometimes lasting up to several months, a wind tunnel experiment may measure close to 50,000 points, each recording up to a thousand variables. Numerical model analyses employ techniques such as computational fluid dynamics to determine these physical performance properties from virtual designs (e.g., CAD drawings.) Numerical solutions are computationally resource intensive and can generate gigabyte-size results.

DARWIN provides not only distributed, near-real–time remote access to large volumes of data, but also tools for data analysis, visualization, and collaboration. DARWIN deals with such "real-world" issues as security requirements and the semantic inconsistency endemic to extending legacy systems (i.e. naming conventions and variations in meaning).

The DARWIN system allows its users to access aerospace data through a collection of displays and to perform various analysis functions on that data. In a typical session, a DARWIN user could perform the following tasks:

- **Establish security context.** On connecting to the DARWIN web server, the user logs in with her name and password. Our facility serves a national community of aerospace designers. Such customers are unenthusiastic about granting competitors access to data on their proprietary designs. To address these concerns, all communications with the web server take place over secure http. Users are authenticated not only by password, but also by IP address.
- **Browse.** The DARWIN home page provides overview screens for the available wind tunnel

tests and computational fluid dynamics solutions. The only tests the user sees are those for which he or she is authorized. From these screens the user can see which tests are in the system, get basic information about those tests, and check the test's bulletin board for messages and related files. Figure 1 shows an example DARWIN home page with wind tunnel tests displayed.



Experimental Data

System Status

DARWIN system is currently up and running.

User **jdwalton3** has access to the following tests:

40 X 80 Foot Wind Tunnel

Test 1903 in progress (click tunnel icon for live update)

Review	Test Number	Test Title	Start Date	Info	Notes
	1903	Wright Flyer Test	3/1/1999		

9x7-Foot Supersonic Wind Tunnel

Review	Test Number	Test Title	Start Date	Info	Notes
	210197	Oblique All Wing Cooperative Test	6/1/1994		

Figure 1: DARWIN Home page with wind tunnel tests displayed.

An in-depth look at the data can be performed by creating a dataset *review*. The user selects the data of interest via the browsing screens and launches the review screen. The review consists of two types of tabular displays (data summary table and sequence table) and a set of plots. The user can choose which variables are displayed in the tables and in the plots. Additional data points can be added to a review by invoking the query screen and searching for points of interest. Figure 2 shows a three-dimensional plot of some wind-tunnel data.

Having selected the data to review and configured the tables and plots, the user can save her work into a DARWIN *dataset*. This structure retains enough information for DARWIN to recreate the review screen at a later time. Datasets are managed in a database on a per user basis.

- **Interact with the world.** DARWIN is not just a database retrieval and presentation mechanism. It also provides two kinds of interaction: real-time monitoring of in-progress testing and collaboration with colleagues. While a wind tunnel test is in progress, users can monitor its progress via the *live screen*. The live screen has the same tables and plots as the review plus current status indicators, message board, and a shared files "shelf." The tables and plots are updated every 20 seconds to show the latest collected data. Figure 3 shows the live screen in operation.

Wind tunnel test data are grouped into "time instants" or *points*. For each point, the wind tunnel data acquisition system collects a large amount of information about conditions in the tunnel and on the model. For example, pressure taps on the model can reveal detail about the flow characteristics at specific regions on the model, and pressure sensitive paint can produce a continuous pressure surface map. Numeric data is stored on the file system, and pressure sensitive paint results are recorded with a camera and stored in image files. DARWIN deals with a variety of file types, both text and binary.

Wind tunnel test data are hierarchical. The set of measurements about a model are a *test*. A given test can be checked for different *configurations* (e.g., orientation of the model and specification and arrangement of sensors), a given configuration can be checked for a specific *run*, and for a run, data is collected at points. There is data associated with each of these levels; points have both the greatest volume and least regular data.

The large volume of data associated with actual tests has led to a design incorporating a meta-database. This database is a relational database that stores information *about* the test, configurations, runs and points. The meta-database holds both data applicable to the experiment as a whole and variables on which users are likely to want to search. For example, tunnel conditions such as wind speed, temperature and angle of attack of the model are data applicable to the entire experiment. Likewise, overall lift and drag apply to the model as a whole. Because users may want to find, for example, the data point with the greatest drag, that information is also included in the meta-database. The data contained in files produced by specialized measurement systems are considered detail data. The meta-database stores the locations of these files, and the time points with which they are associated. However, to display this data, the interface needs to retrieve the actual file.

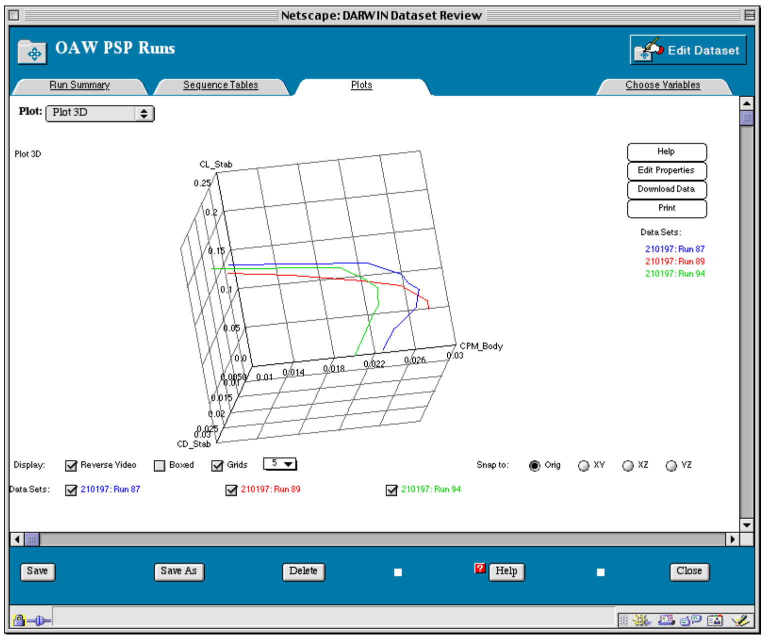


Figure 2: The DARWIN Review Screen showing a 3-dimensional plot

Although DARWIN is an aerospace application, it is also a generic application. What DARWIN does is (1) present to distributed users large volumes of both numeric and image data gathered from multiple sources and (2) provide visualization tools for examining the data, collaboration tools for working cooperatively with the data, and real-time mechanisms for interacting with ongoing activities. The DARWIN architecture and experience thus generalizes across domains with similar (and simpler) problems. One of the goals of the D3 development is to enable to application of the DARWIN-like systems to domains besides aerospace vehicle design.

Design of D3

Critical requirements of D3 include the desire to perform complex interactive visualizations on the client, to allow users to incorporate their own visualizations of data, to access and download select portions of the raw data, to avoid the delay associated with dynamically downloading applets, to overcome the interface limitations of browsers, and to enforce access control rules for data and studies. To understand these issues, it helps to begin by elaborating the D3 meta-data model.

The D3 meta-data model is made up of models, tests, configurations, runs and points. These are maintained in a database as relational tables, where the row identifier (primary key) of a test is a foreign key in the configuration table, the row identifier of a configuration a foreign key of the run table, and so forth. While there is some commonality among tests about what the other columns of these tables ought to be, there is sufficient variety in the meta-information needed about points that many point attributes are stored in a table of <point-id, property-name, value> triples. The actual data archived from a wind tunnel test is also available for access and analysis by the user; the location and display attributes of the raw data are stored in the meta-data model. A good example of this is the pressure tap data. For almost all wind tunnel tests, pressure taps are used to measure the pressure on the aircraft at given physical points. This data is stored as an uncalibrated set of analog measurements in a file. The actual calibration file is also stored, and the location of the pressure tap is stored in a third file. All that is stored in the D3 data model is the time instant a instrument-type of “pressure tap” was recorded, the file locations of the data, and the particular protocol used to access the remote files. The “business logic” of D3 is required to know that (based upon the instrument-type) the three files need to be retrieved, parsed, integrated, and display in a default style.

At this point D3 uses another model to manage “reviews” of the data. These “reviews” can be thought of as similar to the concept of database views except that in the instance of D3/DARWIN, not all of the data resides in a relational database. Another aspect of reviews are that they can be collected and personalized into “studies”. Studies are objects unto themselves, which can have arbitrary relationships with other studies and external files. (The current organization of studies is in a classical directory/file tree, though we plan to experiment with other organizations.) Client applications want to see this organization of data, and can use the data in unpredictable ways—for example, displaying a table of particular attributes of a set of configurations, or plotting the results at selected points. In D3, we are integrating CORE, the next generation of PostDoc [1], an in-house developed document management system. This system that already manages documents, URL’s and data files in a hierarchical directory structure will be used to similarly manage and share the D3 studies and reviews. CORE is discussed below.

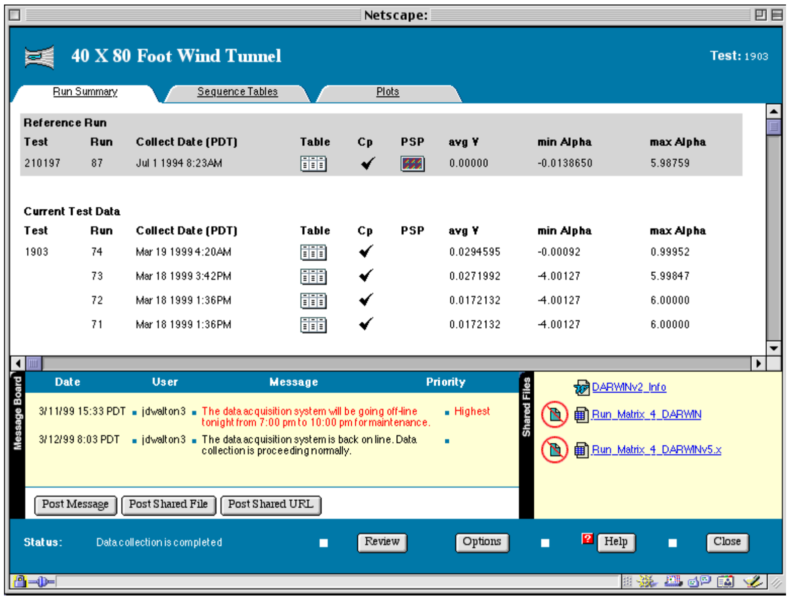
Access control for wind tunnel data is currently centered on tests. That is, one can see all configuration, run and point data about a test if and only if one is allowed access to that test. However, the data for all tests are mixed in the same tables. Security requirements therefore dictate that client programs cannot be allowed to run random queries at the test, configuration, run and point tables. Studies must inherit the arbitrary access control relationships of the meta-data.

The most straightforward way to approach this problem is to give the client application the illusion of having resident the (access-control-limited) wind tunnel data, and arranging "behind the scenes" to fetch the data from the server as needed. Wind tunnel data is read-only; studies and reviews can be both read and written. In D3, the client keeps a cache of the information it has learned. The communication between the client and server is phrased in terms of general queries and responses that include collections of arbitrary "facts" to be filled into the client objects in anticipation of their use.

This behind-the-scenes fetching needs to be more efficient than getting single items at a time. To display a table of ten attributes of twenty items, we don't want to make 200 calls from the client to the server. Requests need to be generated that anticipate the data requirements of the client. Examples of mechanisms for doing this would include having the client explicitly ask for the data needed ("I know I'm filling in this table, so I'll get all the data for the table at once."), provide the context of a request ("Here's the stuff I'm working on now: the objects and columns. Give me what I need.") or having the server "learn" the access patterns of clients ("In this situation, what's usually asked for next is the following. Send it ahead.")

Currently the server logic has a straightforward job. Its function is to get data from the databases and remote file stores in response to client queries, integrate and or synthesize client data, vet it against the permissions of the particular user, and encode it for transmission to the client. Since vetting is required, client requests are made in terms of the "full external name" of objects (e.g., "test 43, configuration 12, run 5"). The server keeps a cache that maps such names to primary keys (that is, in this example, the row identifier of that run). Elements in the cache for a particular user need no other security; elements not in the cache are vetted for access. (This mechanism also allows the client to save its state in whatever way it likes, using the full names of objects to restore another session.) Eventually, the server will have to manage the "distributed data" problem of queries that can refer to more than one database at more than one location, and will needed expended logic to handle more complex tasks, such as the instigation of a complementary numerical analysis when appropriate conditions are met in an experimental test.

The overall architecture of D3 is illustrated in Figure 4: a client that sends requests to a server. That server communicates with a variety of backend elements: a database of wind tunnel test results, a file system of elements such as pressure-sensitive-paint images or data files, a meta-database that points to these files, an LDAP server storing user authentication and access privilege information, and a collaborative system for sharing "studies." What's interesting about this architecture is both its resemblance to classical three-tiered internet applications and its differences. The classic three-tiered architecture has (1) a thin client that communicates with a (2) web server; the web server retrieves and stores information on (3) the database. The server sends HTML (or XML) pages to the client. The client is just an interpreter of the "commands" on these pages. The server incorporates the "business logic" of the application. In the D3 model both the server and the client have the "business logic", distributed to leverage client-side processing. Additionally, D3 has an additional layer below the database, where multiple data servers may retain control and storage of the raw data files. The client is thick, and drives the conversation based on its needs. The client may request multiple files from multiple data servers, the database and the collaborative systems, and display them however it likes.



Reference Run

Test	Run	Collect Date (PDT)	Table	Cp	PSP	avg Y	min Alpha	max Alpha
210197	87	Jul 1 1994 8:23AM	▦	✓	▨	0.00000	-0.0138650	5.98759

Current Test Data

Test	Run	Collect Date (PDT)	Table	Cp	PSP	avg Y	min Alpha	max Alpha
1903	74	Mar 19 1999 4:20AM	▦	✓		0.0294595	-0.00092	0.99952
	73	Mar 18 1999 3:42PM	▦	✓		0.0271992	-4.00127	5.99847
	72	Mar 18 1999 1:36PM	▦	✓		0.0172132	-4.00127	6.00000
	71	Mar 18 1999 1:36PM	▦	✓		0.0172132	-4.00127	6.00000

Message Board

Date	User	Message	Priority
3/11/99 15:33 PDT	jdwalton3	The data acquisition system will be going off-line tonight from 7:00 pm to 10:00 pm for maintenance.	Highest
3/12/99 8:03 PDT	jdwalton3	The data acquisition system is back on line. Data collection is proceeding normally.	

Shared Files
DARWINv2_Info
Run_Matrix_4_DARWIN
Run_Matrix_4_DARWINv5.x

Status: Data collection is completed

Figure 3: Live Screen. DARWIN also provides several collaboration mechanisms for keeping in touch with team members. Users can post messages and files associated with a particular test and can define mail groups for sending group email and tracking the threaded conversations.

The current fashion is to implement the server in a three-tier architecture these days as an Enterprise Java Beans server, relying on EJB to manage the persistence of “bean objects;” retrieving them from the database, caching them on the server, and even providing a public persona for them.

We investigated the EJB approach but rejected it for now. EJB seems particularly appropriate for applications that deal with thin clients (for example, a web browser), handle a row of the database at a time (for example, the description of a single item in a commercial catalog), where many rows are of common interest (for example, when many people want to order the same thing), where there is some useful processing to be done on the server (for example, checking the order consistency and recommending other things to buy). Our thick client needs lots of data (for example, examining a spreadsheet-sized data collection or graphical numeric visualization), can’t be allowed external access to data objects without vetting, and needs to have data sent to it in anticipation of future requests. On the other hand, having passed the data to the client, the server could be less likely to need it again (The client is keeping its own cache. It remains to be seen if simultaneous examination of the same wind tunnel data by different clients will merit a server-side data cache.) Relying on the EJB server to cache all of the user requested data on the server could thus be an unneeded overhead. Assessment of the D3 prototype usage will help determine the requirements for server caching.

Core

CORE [4] (Complex Object Relationship Engine) is the next generation of PostDoc. It is designed to handle continuous and incremental user enhancement of knowledge data, such as metadata about stored documents and archived mailing list communication, user extension of the data model, and, for workflow applications, user contributed application logic.

A key feature of CORE is advanced access control and authentica-authentication mechanisms. Databases typically partition access control on the table, object, or similar large scale. This level of access control is insufficient in an environment where the atom of manipulation by collaborators may be individual metadata attributes. For example, a document may have a variety of attached attributes, such as *owner*, *name*, *keywords*, *description*, and *related documents*. In a review process, an additional attribute of “review rating” may be added/manipulated by the reviewers of the document, who may have privileges to manipulate the document’s other attributes. The important element here is to provide variable granularity—control at the appropriate level for each kind of data.

Few open standards exist for access control and management architectures. However, an emerging standard has been established by the Web Distributed Authoring and Versioning [9] Working Group that provides the level of control and flexibility required by the redesigned PostDoc environment. Also, WebDAV is an excellent target communication mechanism: the standards being developed are for client-server communications for web servers.

Related Work

Within NASA several systems have been developed with remote access to aerospace data. In 1994, NASA Ames Research Center developed a system called remote access wind tunnel (RAWT) [5]. In 1995, NASA Glenn Research Center modified this concept to create remote access control room (RACR). Both systems were Unix only and used a commercial whiteboard program called InPerson for Silicon Graphics machines and X-windows. No database of information was developed, as the emphasis was remote access and collaboration.

Similarly, several data and documentation tools

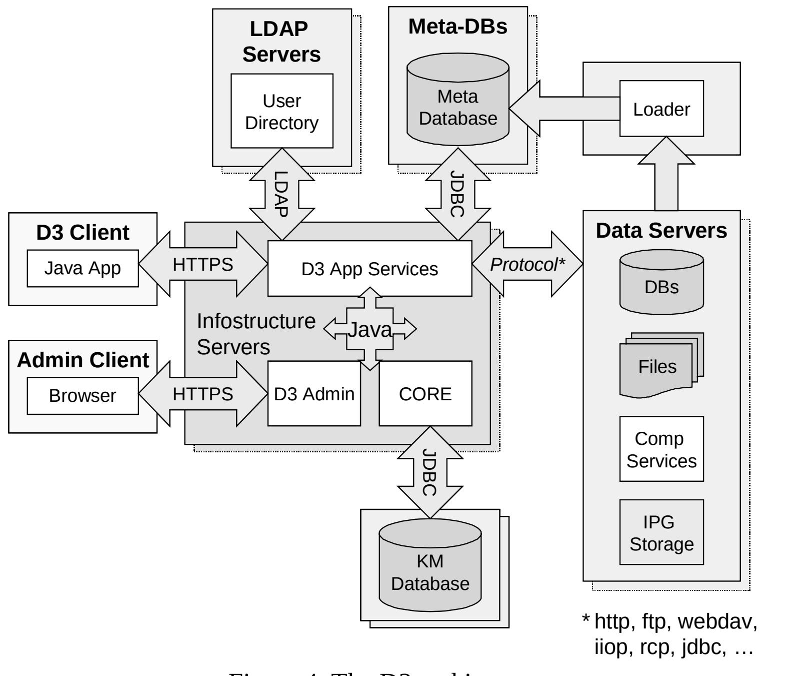


Figure 4. The D3 architecture.

were developed at NASA Langley Research Center and NASA Ames. These were PrISM, and ADAPT at Langley and PostDoc [1] at Ames. PrISM collected the wind tunnel data into a database on a test by test basis and provided a robust query capability to download the results to the user. No presentation or naming consistency was developed under PrISM. ADAPT and PostDoc were similar in that they were early web-based document management systems. ADAPT focused upon creating secure access to encrypted documents. Any data or documents were stored as an encrypted file with the user required to have a decryption helper application for the web browser to decrypt data to the desktop. PostDoc emphasized capturing and translating documents into Adobe PDF files to broaden access to many platforms. PostDoc originally only addressed security through a user identifier and password scheme. It now includes transport layer encryption over HTTPS connections.

Of course, use of the Internet for database access, collaboration, and real-time monitoring has many antecedents. We mention four examples. Evans and Rogers [2] report on using Java applets and CORBA to reimplement (parts of) an existing multi-user WWW application, replacing the existing CGI scripts. They found the applet/CORBA combination to be better at performing client-side applications, to be easier to maintain, to be simpler to program (because of the ability to retain server-side state), to be more straight forward to deploy, and to provide greater responsiveness.

Ly [6] describes Netmosphere ActionPlan, a web-enabled project management product. Architecturally, ActionPlan is a pair of client applets linked to a Java-language server. A key element of Netmosphere was the real-time, selective notification mechanism for keeping information synchronized.

Itschner, Pommerell and Rutishauser [3] report on the GLASS system, which uses internet technology to monitor remote embedded systems. GLASS proxies accumulate data from embedded system monitoring devices and store this information on the database of a server. Client applications, running in browsers with Java applets, retrieve this data through CGI scripts on the server.

Tesoriero and Zelkowitz [7] have developed the WebME system, which uses a mediating query processor, metadata database, and wrappers on the information repositories to direct queries to the appropriate databases.

Closing Remarks

We have presented the DARWIN, a system for the collaborative use of wind tunnel and computational data in aerodynamic design. We are currently constructing D3, the third generation of the DARWIN system, based on a multi-tier model involving a “thick” client, layers of data access, and complex access-control on the server. Critical issues in this development include efficiently moving large amounts of data from the data stores to the client, ensuring the access-control rules are followed, and providing the users with an appropriate collaborative environment.

Acknowledgments

The authors would like to thank the DARWIN development team for building quality applications and making the project a success.

References

[1] Becerra-Fernandez, I., Stewart, H., Del Alto, M., and Knight, C. Developing an Advanced Environment for Collaborative Computing. Proc. Twelfth International Florida Artificial Intelligence Research Symposium, Orlando, Florida, Menlo Park: AAAI Press, May 2000, pp. 154-158.

[2] Evans, E. and Rogers, D. Using Java Applets and CORBA for Multi-User Distributed Applications. *IEEE Internet Computing 1,* 3 (May 1997) 43-55.

[3] Itschner, R., Pommerell, C., and Rutishauser, M. GLASS: Remote Monitoring of Embedded Systems in Power Engineering. *IEEE Internet Computing 2,* 3 (May 1998) 46-52.

[4] Knight, C. and Aha, D. A Common Knowledge Framework and Lessons Learned Module. in D. Aha and R. Weber (Eds.) *Intelligent Lessons Learned Systems: Papers from the AAAI Workshop,* Technical Report Technical Report WS-00-03, Menlo Park: AAAI Press, 2000, pp. 25–28.

[5] Koga, D. J., Schreiner, J. A., Buning, P. G., Gilbaugh, B. L., and George, M. W. Integration of Numerical and Experimental Wind Tunnel (IofNEWT) and Remote Access Wind Tunnel (RAWT) Programs of NASA. 19th AIAA Advanced Measurement and Ground Testing Technology Conference, New Orleans, LA, June 1996, AIAA-96-2248.

[6] Ly, E. Distributed Java Applets for Project Management on the Web. *IEEE Internet Computing 1,* 3 (1997) 21-27.

[7] Tesoriero, R., and Zelkowitz, M. A Web-based Tool for Data Analysis and Presentation. *IEEE Internet Computing 2,* 5 (September 1998) 63-69.

[8] Walton, J., Filman, R. E., and Korsmeyer, D. J. “The Evolution of the DARWIN System.” 2000 ACM Symposium on Applied Computing, March 2000, Como, Italy, pp. 971-977.

[9] Web Distributed Authoring and Versioning (WebDAV) Resources. 2000, http://www.webdav.org